**Chapter 4: Experimental Framework**

The research carried out is a four phase project. The first phase involves extraction of the features from the speech audio samples. This also involves making context windows out of the frames of the audio speech sample and calculating the mean and variance of several consecutive context frames to further capture the fluctuations in the signal.The features are also normalized to ensure every sample contributes equally. The second phase involves feature selection among the extracted features for collecting the best contributing features and inputting these features to the Initial Neural network. The Initial Neural network is then trained with the selected features. The third phase involves applying feature selection among all possible pairs of languages under consideration so as to extract the best contributing features for every pair. These features are then inputted to the Binary Neural Network and the Binary Neural Network is trained. The fourth and the final phase involves testing the samples against the Hybrid Neural Network. The third and the fourth phase are intertwined in the sense that the training and testing data are split according to  
K Fold Cross Validation which makes efficient use of data. The samples are divided for training and testing purposes. Features are extracted from the test samples. This also involves making context windows out of the frames and calculating the mean and variance of several consecutive context frames. The features are then normalized and the same features during the training/second phase are selected and then inputted to the Initial Neural Network. This network generates the top two best candidates for the given sample. These candidates are provided to the appropriate Binary Neural Network which selects the same features during the second training/third phase. These features are focused on classification between the two given languages. These features are then inputted to the Binary Neural Network which provides the final output.

**1) Feature Extraction**

**Dataset Description**

The data set used for the purpose of this project was extracted from https://www.audio-lingua.eu/. A number of different sources were tried but the most promising results were obtained by using the data from the above mentioned website. The site contains large number of 16kHz frequency, mono audio speech samples for both male and female voices in a variety of languages. The clips do not have a fixed length and vary in their duration greatly. We created a script which automatically opened up the browser and loaded a specified page. It then copied the link to download each .mp3 provided on the site till a certain specified number. These links were collected into a file and were used to download the files into appropriate folders. These .mp3 files were converted into .wav format which was suitable for feature extraction. We scraped the recordings of 3 languages: Chinese, French and German. A total of 200 samples were downloaded for each language to be used for both training and testing purposes.

**Extraction of Features**

We experimented with using a number of features for classification but the most promising results were obtained by using the mfcc features along with delta mfcc and delta deltamfcc features. This resulted in 39 features for each frame. The mel-frequency cepstrum is a representation of an audio signal on the mel scale, a nonlinear mapping of frequencies that down-samples higher frequencies to imitate the human ear’s ability to process sound. In our implementations, we used the first 13 cepstral coefficients as our primary features, as is common in similar applications. Delta features and the Delta Delta Features are the first and second time derivatives of the cepstral coefficients, capturing the change of the cepstralfeatures over time, which are useful in classifying language, since pace is animportant factor in language recognition by humans. We calculated these features as the central finite difference approximation of these derivatives. The audio was divided into frames and the features were extracted using 400 frames in a single window with 100 frames overlap between the windows.

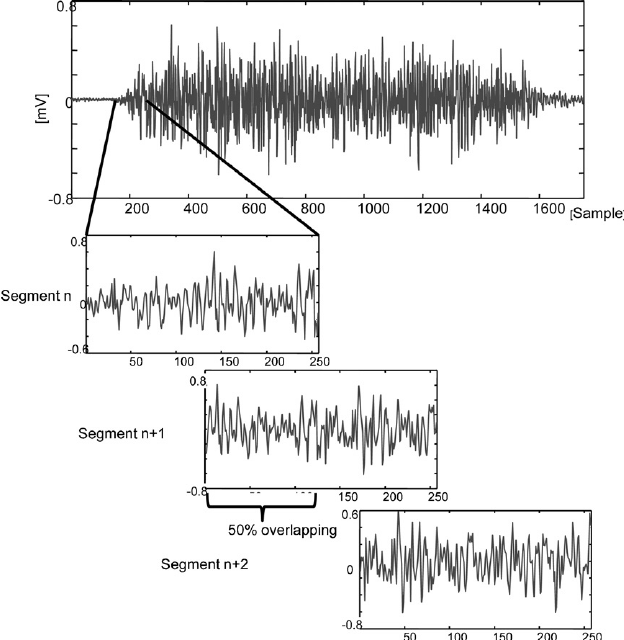
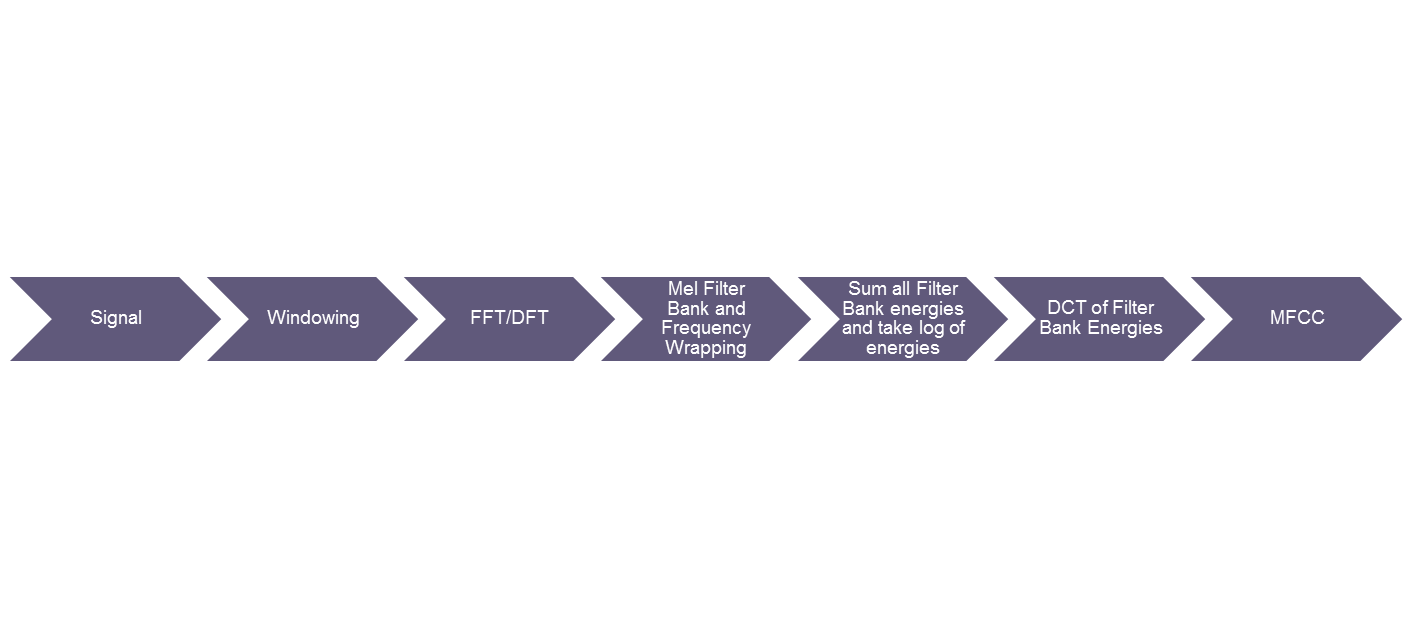


Fig **{number}**Sliding Window

The 13 mfcc features were extracted using pyAudioAnalysis**[cite]**. We tweaked the code for the library to extract only the required 13 mfcc features rather than extracting all the frequency and time domain features to reduce the processing time. Delta and Delta Delta features were calculated using LibROSA**[cite]** which used 3 frames to calculate the estimation of Delta and Delta Delta. We used a context window of size 5 to capture the fluctuations in the speech sample. These were made by stacking the 5 frames after the current frame to make a single feature vector. Average Windows were made after this by specifying the average frames per sample and breaking the audio into equal windows of this size. Mean and Variance for all the features in each feature vector was calculated by taking all the average windows into consideration. This resulted in a feature vector of length 78. This feature vector was then normalized using standard normalization using mu and sigma calculated on the entire test data.

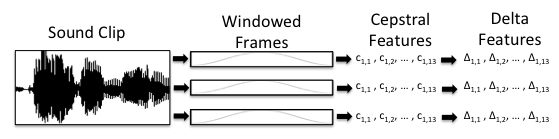


Figure **{number}**Feature Extraction Process

**Preprocessing of Data**

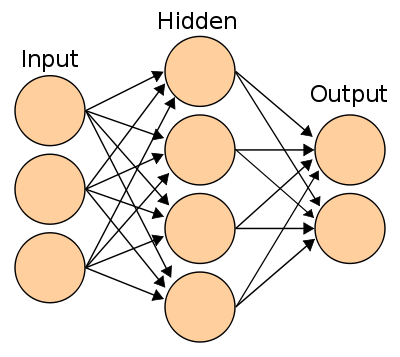
Since the recordings were of variables size we transformed the data by splitting each file such that we obtain a large number of equal duration files for consistency during the training as well the testing phase and also equalize the contribution from each recording. We obtained 1341 samples which were used for both training and testing. Since learning was supervised the labels for each language were to be specified. This was done by organizing the data into directories and while splitting each file the label was specified according to the directory.

**2) Training the Initial Neural Network**

Feature Selection was applied to the extracted features to extract a total of 180 features from each context frame. The features were ranked on the basis of chi squared statistics and the features with the k highest scores were selected, k being the number of features to be selected. The selected features are provided to the Baseline SGD Classifier as well as the Initial Neural Network. The SGD Classifier was implemented using the scikit-learn**[cite]** library and required only calling the fit method to train the model. Since the training was required to be in batches, warm start parameter which remembers the model trained previously was set to True. The Initial Neural network was implemented using Keras**[cite]** which provide a higher level of abstraction on top of the theano**[cite]** backend. The underlying neural network consisted of 3 fully connected layers with the first layer called the input layer containing neurons equal to the selected number of features i.e 180. The second layer called hidden layer contained 12 neurons which were determined by extensive tuning of the model. The third layer called the output layer contained neurons equal to the number of languages to be identified which are 3 in this project. The associated activation function for each of the neurons in the input and hidden layer is relu which stands for rectifier liner unit. The activation function used for the output layer is softmax which produces probability density as its output. The weights were initialized using a uniform initialization scheme and adadelta optimizer was used to control the learning rate. This optimizer makes the learning independent of the initial learning rate. Activity regularization L1 and L2 was added at each layer to avoid over fitting. The model aims to minimize the error function which the and is the actual process through which the model learns. Categorical crossentropywas used which is ideal for multiclass classification. The metric that we sought to maximize was accuracy which we found through extensive research was the only ideal metric for Speech related applications. All other parameters were left at default specified by the keras library. The model was trained for 30 epochs using a batch size of 30 samples. The data provided to the Neural Network was labeled data aimed at supervised learning. However, to adhere to the architecture of the Neural Network. The single output label was transformed into a 1 hot vector with the 1 corresponding to the actual spoken language. The training and testing was done using K Fold Cross Validation with K = 10. Therefore, the training data consisted of 9 folds of the total data available. The data samples included in training were further shuffled using sklearn library before fitting the Initial Neural Network to the data. The model changes the weights and bias associated with each neuron in order to minimize the error function. The model generates the best two candidates for further classification by producing probabilities for each language. The two languages with the highest probabilities are specified as the candidates.

**3) Training the Binary Neural Network**

This phase involves feature selection for all possible pairs of languages (excluding the same language pair). This involves applying chi squared statistics on all pairs and selecting k best scoring features. The number of selected features for each pair were found out through tuning of the models and was finally settled at 380. The Binary Neural network was implemented using Keras**[cite]** which provide a higher level of abstraction on top of the theano**[cite]** backend. The underlying neural network consisted of 3 fully connected layers with the first layer called the input layer containing neurons equal to the selected number of features i.e 380. The second layer called hidden layer contained 22 neurons which were determined by extensive tuning of the model. The third layer called the output layer contained neurons equal to the number of languages to be identified which are 2 for the Binary case. The associated activation function for each of the neurons in the input and hidden layer is relu which stands for rectifier liner unit. The activation function used for the output layer is softmax which produces probability density as its output. The weights were initialized using a uniform initialization scheme and adadelta optimizer was used to control the learning rate. Activity regularization L1 and L2 was added at each layer to avoid over fitting. The model aims to minimize the error function which the and is the actual process through which the model learns. Binary crossentropy was used which is ideal for binary classification. The metric that we sought to maximize was accuracy which we found through extensive research was the only ideal metric for Speech related applications. All other parameters were left at default specified by the keras library. The model was trained for 30 epochs using a batch size of 30 samples. The data provided to the Neural Network was labeled data aimed at supervised learning. However, to adhere to the architecture of the Neural Network. The single output label was transformed into a 1 hot vector with the 1 corresponding to the actual spoken language. The labels specified here were consistent with the Initial Neural network. The data samples included in training were further shuffled using sklearn library before fitting the Binary Neural Network to the data. The model changes the weights and bias associated with each neuron in order to minimize the error function. The model generates the final predicted language according to the trained model.

 Fig **{number}** Neural Network Representative Architecture

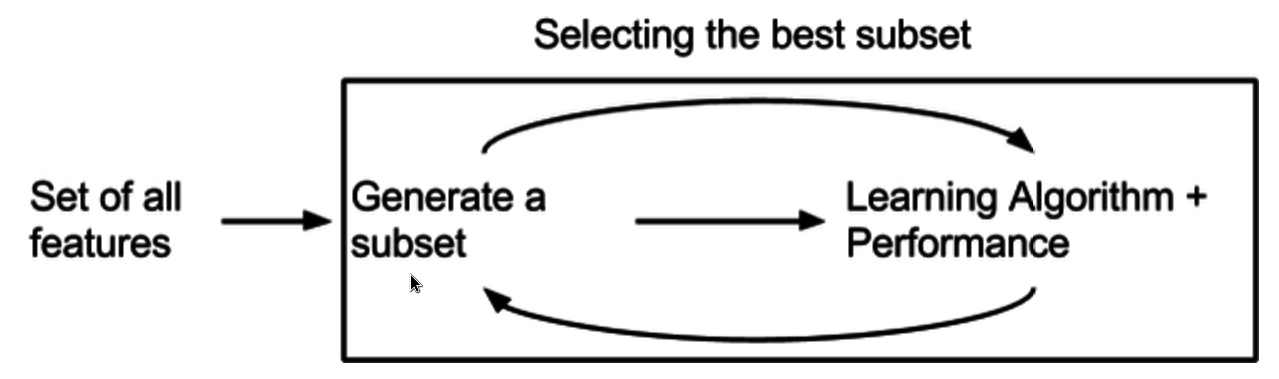
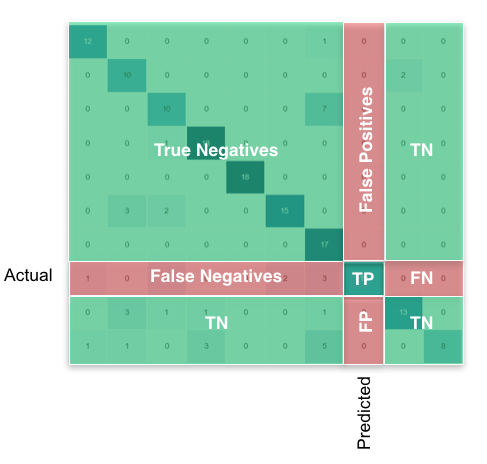


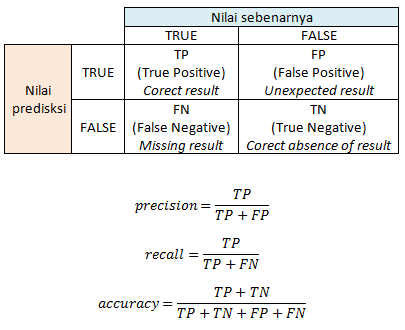
Fig **{number}**Feature Selection Process

Testing

Cross validation is a model evaluation method that is better than residuals. The problem with residual evaluations is that they do not give an indication of how well the learner will do when it is asked to make new predictions for data it has not already seen. One way to overcome this problem is to not use the entire data set when training a learner. Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on ``new'' data. This is the basic idea for a whole class of model evaluation methods called *cross validation*.

The evaluation metrics were Accuracy, Recall and Precision. For a multiclass case it is calculated in the way depicted in the following figure.





We used two types of validation testing.

**Holdout Method**

The holdout method is the simplest kind of cross validation. The data set is separated into two sets, called the training set and the testing set. The function approximator fits a function using the training set only. Then the function approximator is asked to predict the output values for the data in the testing set (it has never seen these output values before). The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the model. The advantage of this method is that it is usually preferable to the residual method and takes no longer to compute. However, its evaluation can have a high variance. The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made.

**In this type of validation we set out 60% of the entire data for training and rest 30% for holdout test. The 30% test samples were independent of training samples, i.e. even the speakers were distinct in both the sample sets.**

**K Fold Validation**

K-fold cross validation is one way to improve over the holdout method. The data set is divided into *k* subsets, and the holdout method is repeated *k* times. Each time, one of the *k* subsets is used as the test set and the other *k-1* subsets are put together to form a training set. Then the average error across all *k* trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set *k-1* times. The variance of the resulting estimate is reduced as *k* is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch *k* times, which means it takes *k* times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set *k* different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.

**While doing K-Fold Validation we used k=10, thus the entire model had to be trained 10 times.**

